Using ssh

$Date: 2006/03/21 16:07:25 $

1 Overview

ssh is a suite of programs that provide a secure replacement to *rlogin*, *rcp*, and *rsh*.

rlogin, *rcp*, and *rsh* (the traditional versions, anyway) suffer a number of security weaknesses. When they prompt for a password, the password gets transmitted across the network as plain text where it can be easily sniffed. And when they don't prompt for a password, they 'trust' the client computer because it's named in '`~/.rhosts`' or '`/etc/hosts.equiv`'; this makes them vulnerable to spoofing.

ssh never transmits passwords - or any other part of a session - as plain text. All network traffic in an *ssh* session is encrypted, providing good protection from sniffers. *ssh* even provides a proxy to encrypt X11 sessions!

With *ssh* hosts are identified via public key encryption in addition to IP address or DNS name, making spoofing much more difficult. (The spoofer has to acquire the target's private key, which is difficult since that's usually encrypted when stored on disk!)

The *ssh* suite consists of a daemon (*sshd*) and a set of user programs:

sshd
: The ssh daemon. This must be running on the server in order for clients to make ssh connections to the server.

ssh-keygen
: Generates public and private keys, which are how ssh replaces '`~/.rhosts`' and '`/etc/hosts.equiv`'.

ssh-agent and *ssh-add*
: *ssh-agent* keeps keys in memory; *ssh-add* adds a key to the list of keys *ssh-agent* is keeping in memory. These let users enter keys once per client session instead of each time they start an ssh session.

ssh, *slogin*, *rlogin*, *rsh*, *scp*, and *rcp*
: These start ssh sessions; these are the most commonly used parts of the package.

ssh-askpass
: ???

This document contains the recipe for a simple *ssh* setup. Enjoy!

See `http://www.tac.nyc.ny.us/~kim/ssh/` and `http://edcissl.cr.usgs.gov/ssh/ssh.html` for more information about *ssh*.

2 Installing ssh

2.1 Obtaining and Installing ssh

The latest officially released version is 1.2.26 (as of July 7 1998).

The central site for distributing the Unix version of *ssh* is `ftp://ftp.cs.hut.fi/pub/ssh/`; it can also be obtained from `ftp://ftp.net.ohio-state.edu/pub/security/ssh/` and `ftp://ftp.gw.com/pub/unix/ssh/`.

ssh has been ported to windows and mac.

The Unix installation is easy. We recommend installing with '`rsaref2`' and running configure as:

```
# ./configure --with-rsaref --with-libwrap
```

Before running configure, you'll probably need to copy '`libwrap.a`' and '`tcpd.h`' from you tcp-wrappers directory into the ssh directory and untar '`rsaref2.tgz`' into the ssh directory.

This configuration will install the daemon in '`/usr/local/sbin/sshd`' and the clients in '`/usr/local/bin/ssh`' etc.

After running `make install`, set up symbolic links to replace rlogin, rsh, and rcp:

```
# cd /usr/local/bin
/usr/local/bin
# ln -s ssh rlogin
# ln -s ssh rsh
# ln -s scp rcp
```

Encourage users to have '`/usr/local/bin`' before '`/usr/bin`' in their path. This way if they run *rlogin*, they'll get the secure version. Even better, encourage users to use *ssh* instead of *rlogin* or *rsh* and *scp* instead of *rcp*. This way they won't accidentaly run the less secure programs when '`/usr/local/bin`' isn't before '`/usr/bin`' in their path!

2.2 Starting the daemon

The ssh daemon, *sshd*, lets the computer be an ssh server. If you only plan to be a client, you can skip this section.

You'll need to restart *sshd* each time the computer reboots:

Solaris 2.5.1

On Solaris, create '`/etc/init.d/sshd`' (or copy it from '`sunland:/usr/local/sbin/etc-init.d-` and then link that to '`/etc/rc3.d/S95sshd`' and '`/etc/rc3.d/K95sshd`':

```
# cd /etc/init.d
# pr -tn  sshd
    1 #!/bin/sh
    2 #
    3 # [Solaris, at least]
    4 #
    5 # This script should get installed as /etc/init.d/sshd,
     6 # /etc/rc3.d/S95sshd, and /etc/rc3.d/K95sshd
     7 #
     8 # tsingle, March 27, 1998
     9
    10 case "$1" in
    11 'start')
    12 if [ -x /usr/local/sbin/sshd ]; then
    13 echo "starting Secure Shell daemon."
    14 /usr/local/sbin/sshd
    15 fi
    16 ;;
    17 'stop')
    18         if [ -f /etc/sshd.pid ]; then
    19 kill `cat /etc/sshd.pid`
    20 fi
    21 ;;
    22 *)
    23 echo "Usage: /etc/init.d/sshd { start | stop }"
    24 ;;
    25 esac
    26 exit 0
# ln /etc/init.d/sshd /etc/rc3.d/S95sshd
# ln /etc/init.d/sshd /etc/rc3.d/K95sshd
#
```

Make sure it is owned by root and not group or other writable.

FreeBSD On FreeBSD, if you install *ssh* from the ports, this gets set up automatically!

3 Client-Side User Accounts

3.1 ~/.ssh and ssh-keygen

The first step in using *ssh* is to run *ssh-keygen* to create your public and private keys and to establish your passphrase. The public and private keys replace '`.rhosts`'; the passphrase makes it difficult for an intruder to steal your private key.

```
vicious$ ssh-keygen
Initializing random number generator...
Generating p: ..............................................................++
(distance 1018)
Generating q:  ..........................................++ (distance 662)
Computing the keys...
Key generation complete.
Enter file in which to save the key (/home/sid/.ssh/identity): <Enter>
Enter passphrase: birdy, birdy in the air ...<Enter>
Enter the same passphrase again: birdy, birdy in the air ...<Enter>
Your identification has been saved in /home/sid/.ssh/identity.
Your public key is:
1024 37 113278418383580367625943265685791983595328734310621075367344063337744648
702902540553328665868583607009399832473243015623453942192750832604739292835155550
36101942533065560596287413968110468291403677498319510886659044129743213898139297
111546096801235608034433814208375391963144299938405642717226859857044580290089 si
d@vicious
Your public key has been saved in /home/sid/.ssh/identity.pub
```

3.2 ssh-agent

ssh-agent makes ssh convenient to use. With *ssh-agent*, you'll only need to enter your passphrase once. After you've entered your passphrase, you'll be able to *ssh* without being prompted for a password.

The trick to using *ssh-agent* is to make it the parent of your shell or X session. Here are three common scenarios:

3.2.1 ssh-agent and a shell

If you're not running an X session, simply start a new shell under *ssh-agent*:

```
$ ssh-agent bash
```

(Substituting, of course, your favorite shell for *bash*).

3.2.2 ssh-agent and xdm

To get *ssh-agent* to be the parent of an xdm session, you'll need to rename your existing '`~/.xsession`' and create a new '`~/.xsession`':

```
$ cd
$ mv .xsession .xsession-real
$ cat > .xsession
$!/bin/ksh
/usr/local/bin/ssh-agent ~/.xsession-real
^D
$ chmod u+x .xsession
```

3.2.3 ssh-agent and CDE

To get *ssh-agent* to be the parent of a CDE session, you'll need (as root) to rename the existing '`/usr/dt/bin/Xsession`' and create a new '`/usr/dt/bin/Xsession`':

```
# cd /usr/dt/bin
/usr/dt/bin
# mv Xsession Xsession-real
# cat > Xsession
#!/bin/ksh
if [ -x /usr/local/bin/ssh-agent -a -d ~/.ssh ]; then
  /usr/local/bin/ssh-agent /usr/dt/bin/Xsession-real
else
  . /usr/dt/bin/Xsession-real
fi
^D
# chmod a+x Xsession
# chmod a-w Xsession
#
```

3.3 ssh-add

Once you've set up *ssh-agent*, you can run *ssh-add. ssh-add* will prompt you for your passphrase and *ssh-agent* will remember that you've entered your passphrase - you'll only need to type your passphrase once per session!

Anyway, here's what an *ssh-add* dialogue looks like:

```
$ ssh-add
Need passphrase for /home/fido/.ssh/identity (fido@alpo).
Enter passphrase: tweet tweet meow meow
Identity added: /home/fido/.ssh/identity (fido@alpo)
$
```

You're almost ready to use *ssh* - all that's left is to set up your account on the server.

4 Server-Side User Accounts

When you ran *ssh-keygen* to create your public and private keys (see Chapter 3 [client-side user accounts], page 4) one of the files that got created was '`~/.ssh/identity.pub`', which contains one really long line that the server can use to identify you via public key encryption.

Cat the client machine's '`~/.ssh/identity.pub`' into the server machine's '`~/.ssh/authorized_keys`' and you're all set to use *ssh*.

'`~/.ssh/authorized_keys`' contains copies of the public keys of every user who can ssh into the account. '`~/.ssh/authorized_keys`' sort-of replaces '`~/.rhosts`'.

5 Using ssh

If you've done the above, using *ssh* is simple.

Instead of '`rlogin coconut`' use '`ssh coconut`'.

Instead of '`rsh coconut w`' use '`ssh coconut w`'.

Instead of '`rcp coconut:juice whitestuff`' use '`scp coconut:juice whitestuff`'.

Got it?